CASE STUDY

Building a Prompt Engineering Toolkit

Uber

1

Introduction

- Crafting effective prompts is key for generating accurate outputs from LLMs (Large Language Models).
- To streamline LLM experimentation, Uber developed a centralized prompt engineering toolkit for building, managing, and executing prompt templates across models.
- The centralized toolkit helps create well-crafted prompt templates, enhanced with context from RAG (Retrieval-Augmented Generation) and runtime feature datasets.
- The toolkit enables prompt creation with system instructions, dynamic contextualization, large-scale batch offline generation (LLM inference), and prompt response evaluation.
- It also supports version control, collaboration, and safety measures, including hallucination checks, a standardized evaluation framework, and a safety policy to ensure responsible AI usage.
- Uber's case study provides an overview of the prompt template lifecycle, toolkit architecture, and its use in production at Uber.

2

Goals

Uber designed the prompt engineering toolkit to enable users to:

- Explore available LLM models.
- Search and discover prompt templates.
- Create and update prompt templates and tune parameters with access controls.
- Iterate on prompt templates through code reviews and version control.
- Generate batch offline generation by applying RAG and testing datasets with different LLM models for LLM inference.
- Evaluate prompts templates via offline evaluation through testing dataset and online evaluation.
- Monitor prompt template performance with LLMs in production to detect regressions and continuously monitor LLM quality through systematic tracking.

3

Prompt Engineering Lifecycle

The case study outlines two stages of the prompt engineering lifecycle: development and productionization, with users engaging with the prompt toolkit at each stage.

Development Stage

This stage includes three phases:

1) **LLM Exploration**: Users explore LLMs through a model catalog and Uber's GenAI Playground. The catalog provides descriptions, metrics, and usage guides for available models, while the playground allows users to test LLM capabilities and their applicability to specific use cases.

2) **Prompt Template Iteration**: Users define business needs, gather sample data, and create, analyze, and refine prompts. Auto-prompting makes suggestions so users don't need to create prompt templates from scratch, while a catalog of existing templates supports reuse. Iterative testing and revisions ensure alignment with objectives.

3) **Evaluation**: Users evaluate prompts against more extensive datasets to assess performance, leveraging either LLM-based evaluation (LLM as a judge) or custom code-based evaluators to ensure prompt effectiveness.

Productionization Stage

In this stage, only prompt templates that meet the evaluation threshold are productionized. Users monitor usage in production, collecting data on system usage to identify opportunities for improvements.

4

Architecture

- The prompt engineering toolkit integrates various components to support LLM deployment, prompt evaluation, and batch inference responses.
- It includes a Prompt Template UI/SDK for managing templates and revisions, connected to GetAPI and ExecuteAPI for interfacing with LLMs in the Model Catalog.
- Templates and models are stored in ETCD and UCS (object configuration storage) and utilized in the Offline Generation and Evaluation Pipelines.

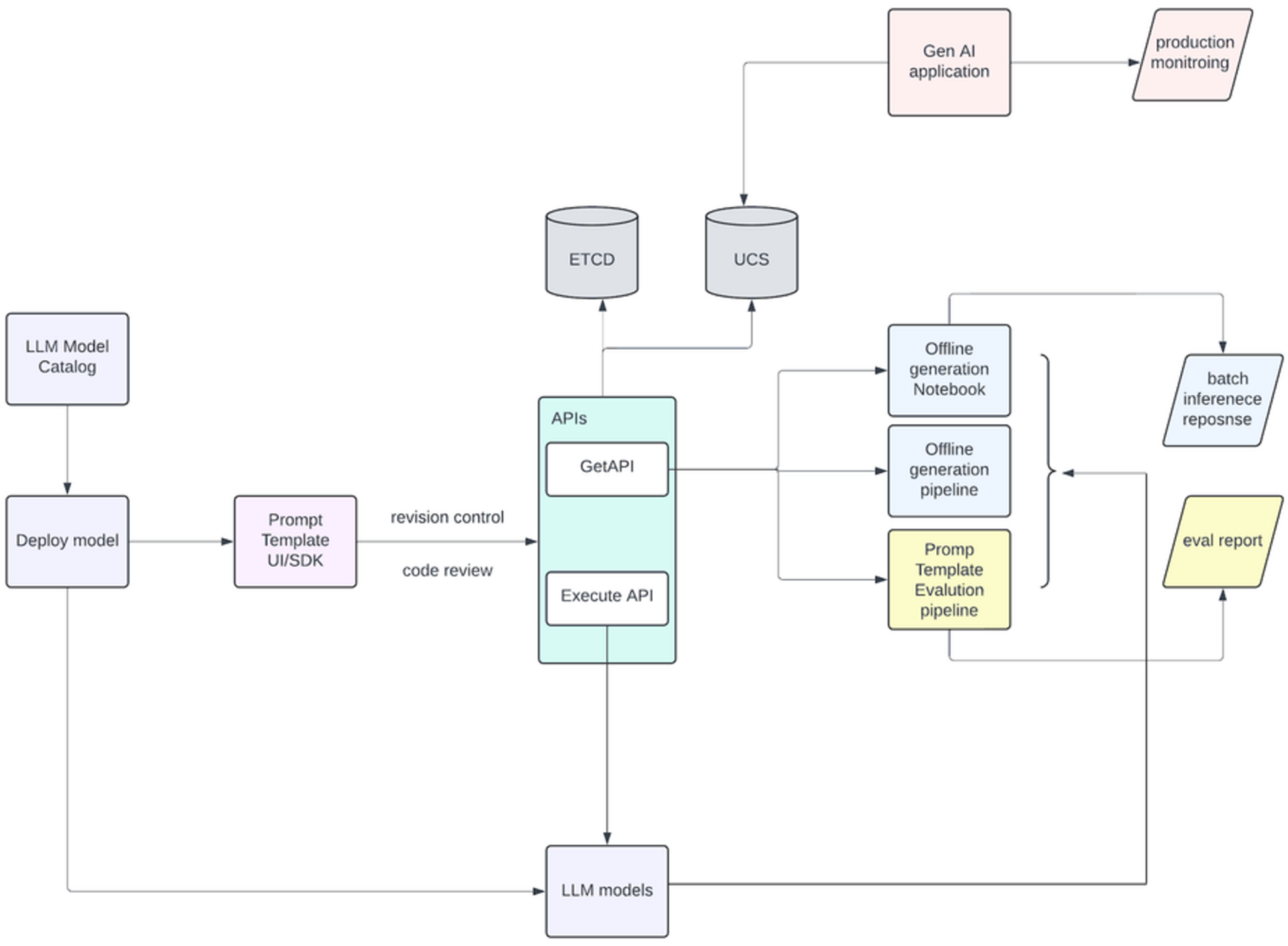

Figure 1: Overview of the prompt engineering toolkit.

5

UI/Design

- Users interact with the prompt toolkit in GenAI Playground, which integrates with the Model Catalog: a repository of Uber's LLMs with specifications, typical use cases, costs, and performance metrics.
- In the playground, users can select models, craft custom prompts, and adjust parameters (e.g. temperature) to evaluate responses during the ideation phase.

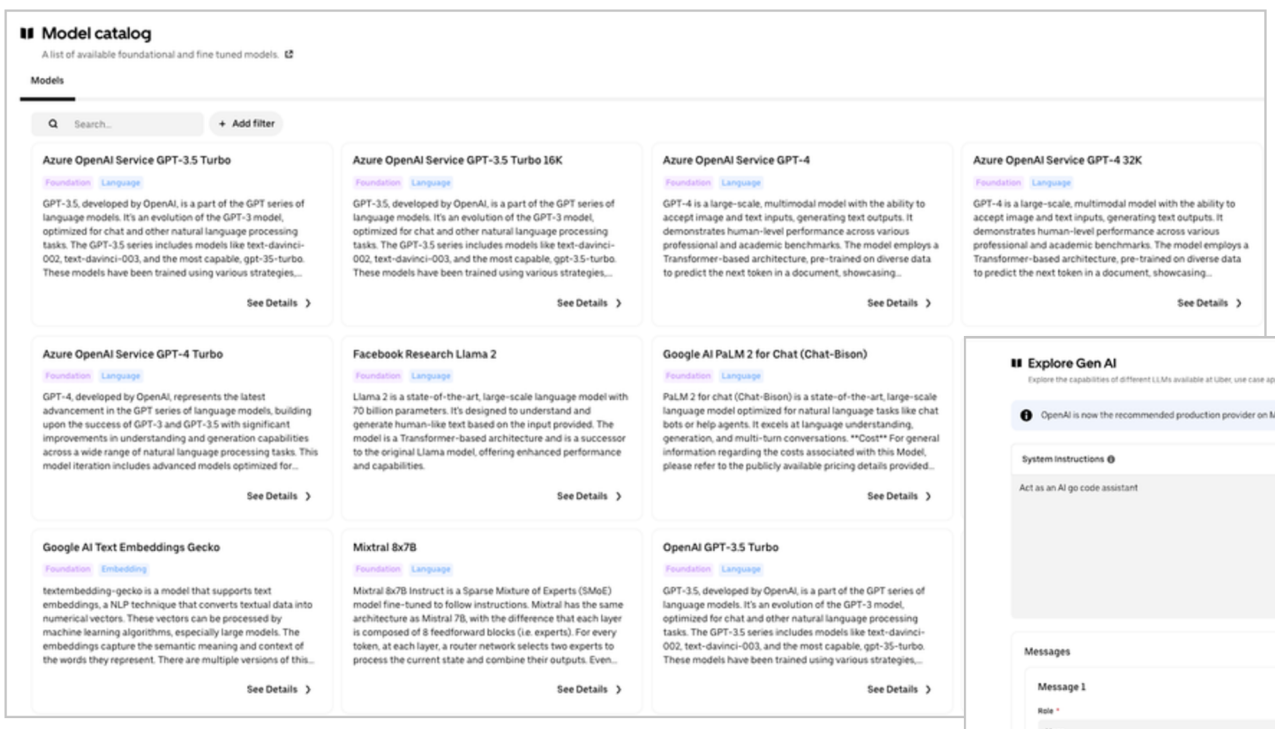

Figure 2: The model catalog UI.

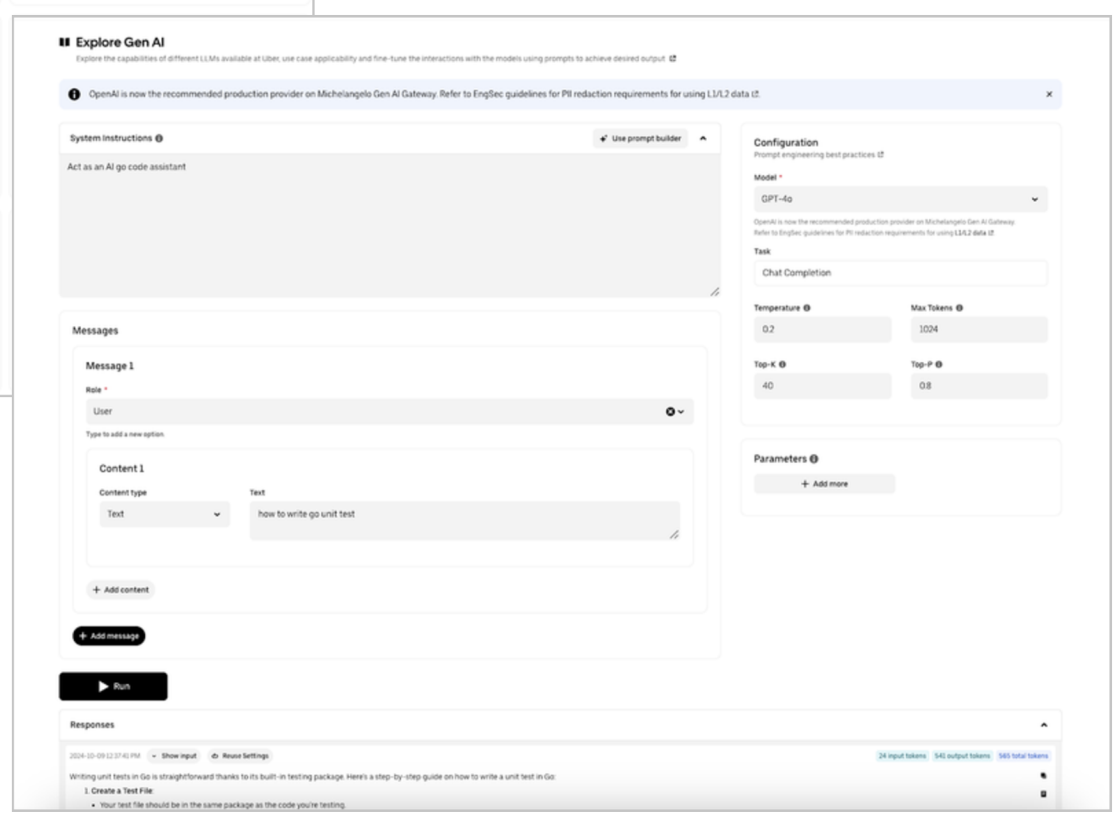

Figure 3: The prompt template edit page.

6

Prompt Template Creation

The prompt engineering toolkit includes an auto-prompt builder that helps users create prompts and discover techniques for their specific use cases. Built on Uber's Langfx framework (an Uber internal Langfx service built on LangChain), it follows these steps:

1. **Best Practices:** Integrates prompt engineering best practices for context RAG retrievers.
2. **Templates and Examples:** Offers templates with detailed instructions and few-shot examples to help LLMs craft user prompts.
3. **LLM-Powered Generation:** Leverages LLMs to auto-generate and suggest tailored prompts.

The prompt builder leverages techniques such as chain of thought (CoT), automatic Chain of Thought, prompt chaining, tree of thought, Automatic prompt engineer, multimodal CoT prompting (see full case study for details on each)

7

Prompt Template
Revision Control

- Prompt template iteration follows review practices. Users modify instructions and model parameters, testing responses with a dataset. Each iteration of a prompt template requires a review, and once approved, a new revision is created.
- To prevent accidental changes in production, users can deploy prompt templates with a specific revision tag. The deployed templates are managed through ObjectConfig, Uber's internal configuration system, and are automatically fetched by the production service upon deployment.

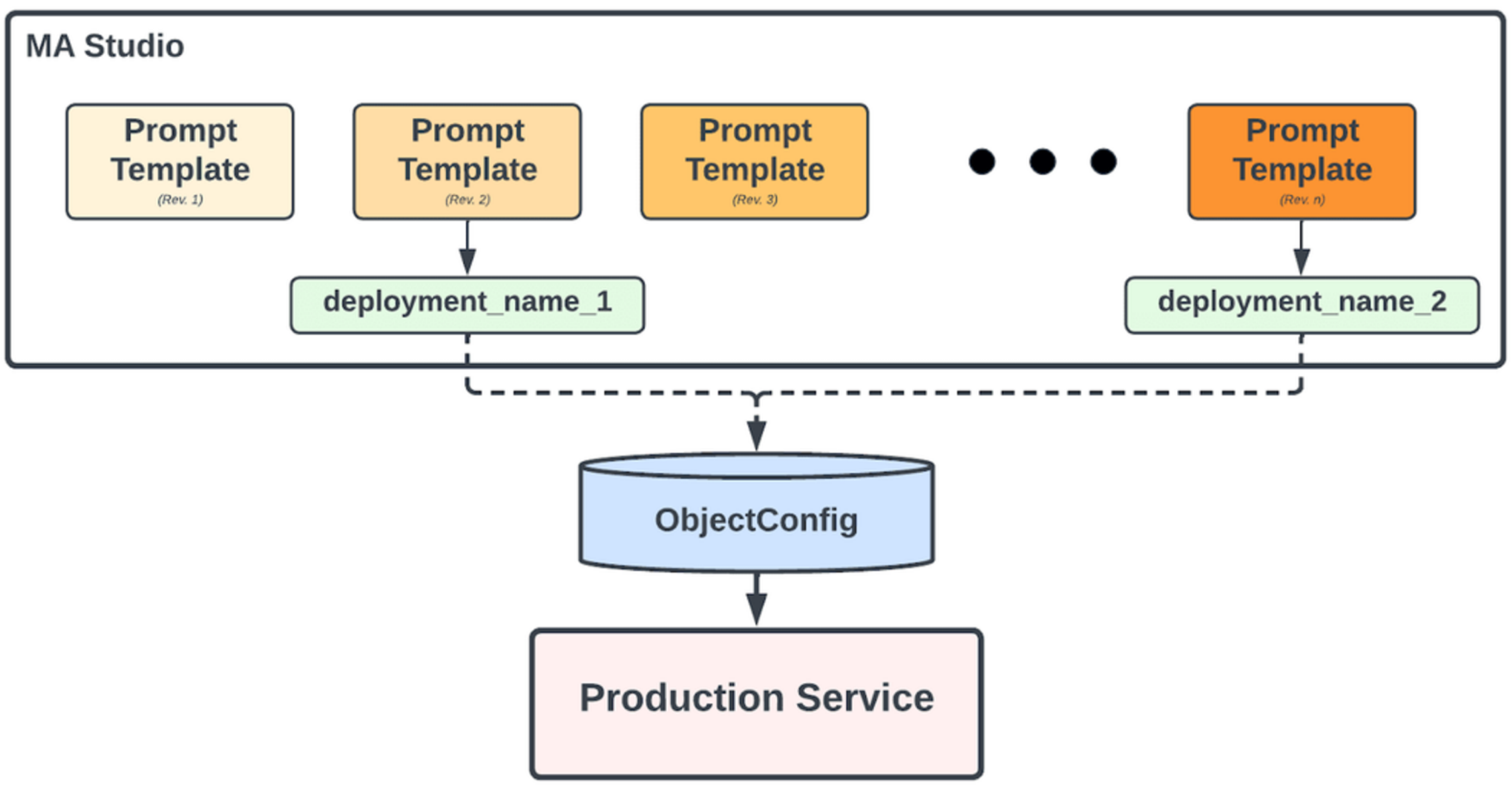

Figure 4: Prompt template revision control.

8

Prompt Template Evaluation

For evaluating prompt templates, two methods are employed by Uber:

1. **LLM as Judge:** Useful for tasks requiring subjective quality, such as generating text that should be engaging, persuasive or stylistically specific.
2. **Custom, user-defined code:** Specific metrics and criteria are coded to evaluate LLM performance, offering particularly tailored evaluation if required.

The toolkit provides evaluation prompt templates, each with instructions, examples, metrics and the format for responses. Templates are evaluated using either a golden dataset with labeled data or a dataset derived from production traffic. Aggregated metrics from large datasets offer a high-level comparison of template effectiveness.

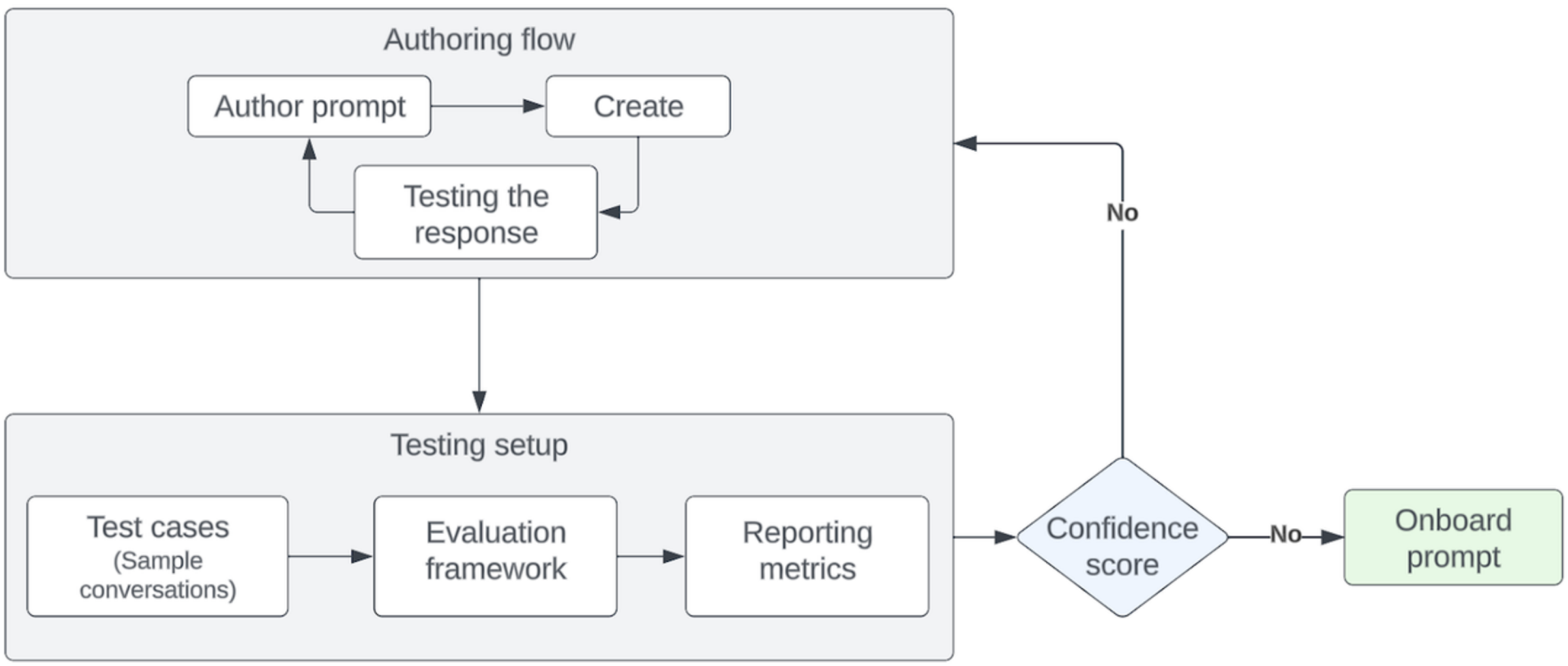

Figure 5: Prompt evaluation flow.

9

Production Use Cases at Uber

Offline LLM Service

- The LLM Batch Offline Generation pipeline enables large-scale LLM inference for tasks like rider name validation, which checks the legitimacy of usernames. This pipeline processes existing and new usernames asynchronously, generating responses in batches.
- In MA Studio (Uber's AI platform), users set up the pipeline by selecting a relevant dataset, which is then dynamically applied to the prompt template. For example, the template "Is this {{user_name}} a valid human name?" is populated with usernames from the dataset, generating custom prompts for each entry.

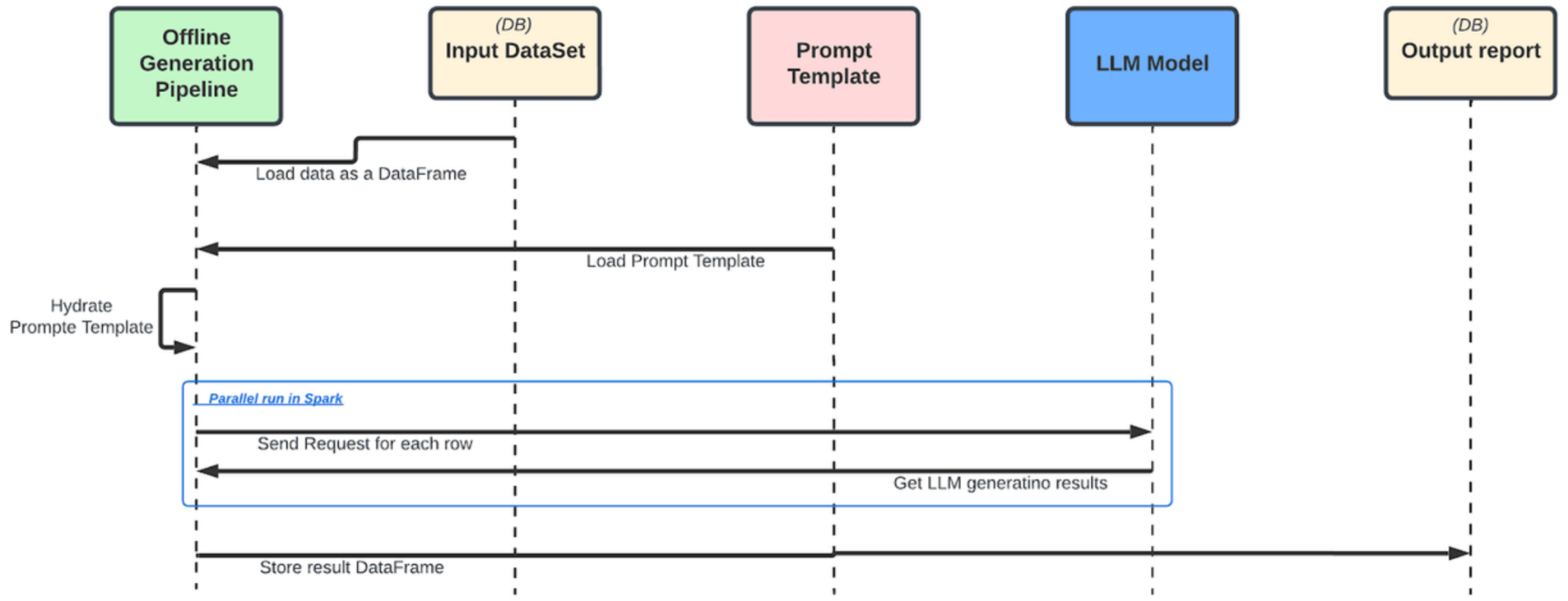

Figure 6: Prompt offline generation.

Production Use Cases at Uber
Online LLM Service

The prompt template contains dynamic placeholders that are substituted with runtime values provided by the caller. Currently, only string-type substitution is supported using Jinja-based template syntax.

The service also supports fan-out capabilities across prompts, templates, and models:

1. **Templates:** The API formats payloads into vendor-specific structures from the generic data model. Multiple templates, such as chat and text completion, are supported.
2. **Prompts and Models:** Prompts are linked to specific models and templates. The service fetches and executes the required parameters via genAI APIs.

For example, in a summarization use case, when a support ticket is handed off to a new agent, the service generates a summary to provide context, eliminating the need for the new agent to ask the customer to repeat the issue.

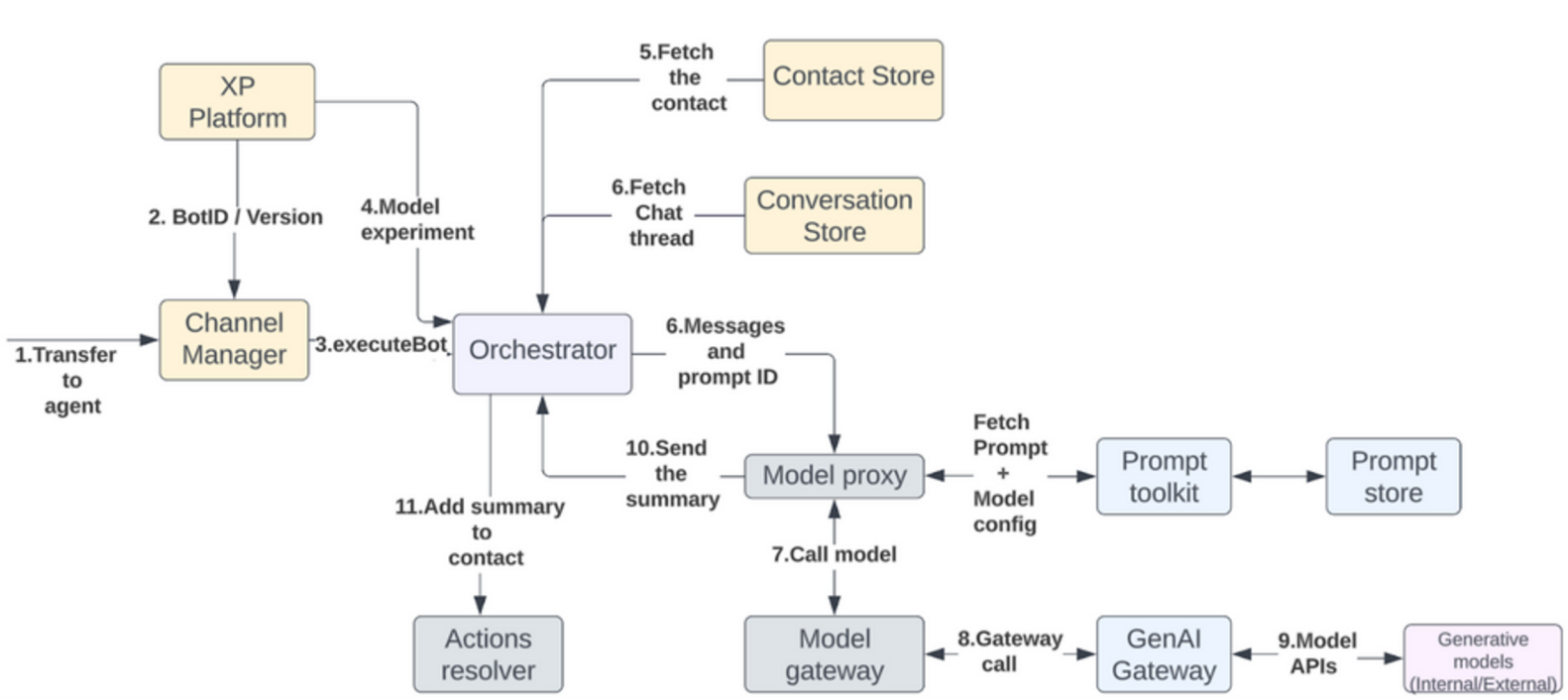

Figure 7: Prompt execution and summarization flow.

11

Conclusion

- The prompt engineering toolkit provides a framework to enhance LLM interactions at Uber, supporting development and production stages.
- From exploring LLM capabilities in the GenAI Playground to creating and iterating prompt templates, it enables users to leverage LLMs effectively.
- The toolkit's architecture ensures a systematic approach to prompt design, combining advanced guidance and evaluation methods for quality results.
- The prompt template lifecycle, from development to production and monitoring, enables rigorous testing and performance optimization.

Attribution

This document summarizes key points from Uber's case study, "Introducing the Prompt Engineering Toolkit" (November 2024). Read the full case study here: https://www.uber.com/en-CA/blog/introducing-the-prompt-engineering-toolkit/

Want to level up with Generative AI? Follow  Lewis Walker

Repost this to help your network